(19) Europäisch s Patentamt
Eur pean Patent Office
Office européen des brevets

(11) Publication number: **0 441 508 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 91300595.5

(51) Int. Cl.5: G06F 12/08

(22) Date of filing: 25.01.91

(30) Priority: 09.02.90 US 477704

(43) Date of publication of application:
14.08.91 Bulletin 91/33

(84) Designated Contracting States:
DE FR GB

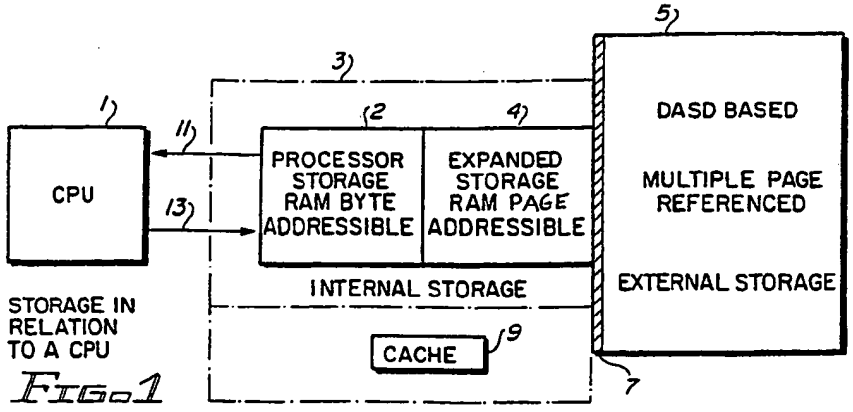
(71) Applicant: International Business Machines Corporation
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: Bozman, Gerald Parks
609 Ramapo Valley Road
Oakland, New Jersey 07436(US)
Inventor: Eisenberger, George
54 Albemarle Road
White Plains, New York 10605(US)
Inventor: Lett, Alexander Stafford
402 MacGregor Drive
Mahopac, New York 10541(US)
Inventor: Myers, James Joseph
950 Columbus Avenue No.6
San Francisco, CA 94133(US)
Inventor: Tetzlaff, William Harold
37 Fox Den Road
Mount Kisco, New York 10549(US)
Inventor: Unger, Jay Harold
Knollwood Court
Mohegan Lake, New York 10547(US)

(74) Representative: Killgren, Neil Arthur
IBM United Kingdom Limited Intellectual Property Department Hursley Park
Winchester Hampshire SO21 2JN(GB)

(54) Data storage using a cache.

(57) Data in pages is mapped into a very large virtual external address space (25) through a cache without disturbing the logical view of the data and without having to assign physical or real backing store to said logical view. A data cache (27) is used in which pages are indexed according to a logical address (23) intermediate to their virtual address and their physical location in external storage (5). Pages common to two or more files are updated in place in the cache, while pages bound to only one file are shadow copied.



STORAGE IN RELATION TO A CPU

Fig-1

EP 0 441 508 A2

DATA STORAGE USING A CACHE

This invention relates generally to data storage and, more particularly, to the type of data storage, sometimes referred to as cache storage, used in conjunction with a central processing unit (CPU) to expedite the processing of instructions.

A typical data processing machine comprises an instruction processor coupled to a hierarchically organised and least recently used (LRU) ordered storage system containing software and data. The fastest, most rapidly accessed storage is positioned closest to the instruction processor. Also, it is placed at the top of the hierarchy. Progressively slower forms of storage, containing the bulk of information, occupy lower positions within the hierarchy.

Because storage costs increase dramatically with speed, many computer systems divide the physical storage subsystem into a number of performance levels. Some of these levels, such as direct access storage devices (DASD) and tape, have been treated as peripheral I/O devices and are accessed over an asynchronous path. Other levels, such as random access memory (RAM) and cache, have been treated directly by system hardware and accessed over a synchronous path as part of internal storage.

The term "internal storage" is customarily applied to that portion of storage randomly addressable for single read or write transfers. In IBM systems, internal storage is byte addressable except for an extension ("expanded store"). Expanded store is randomly accessed on a block or page addressable (4096 bytes/page) basis. It is managed as an LRU real memory backed paging store. Similarly, "external storage" is applied to that bulk portion of storage that is not randomly addressable and must be directly accessed, as on DASD.

An internal store is deemed "synchronous" when a referencing processor idles until a return is received. Generally, if the data being sought resides in external store (beyond a point called the "I/O boundary"), a referencing processor will search for another task to perform instead of waiting. This task switching is disruptive in that a retrieval path must be established to the new data, and the processing state of the prior task must be saved. When the retrieval from external storage has been completed, it is again necessary to switch the CPU back to the former task.

Cache and Cache Invalidate

A "cache" is typically an indexable LRU-ord red collection of pages in a buffer positioned in a path to data or instructions so as to reduce access time. The term "cache invalidate" refers to either removing from the cache or providing indication that a named page is invalid, for example following a change to the base page on some other data path so that the version in the cache is no longer accurate.

A processor or CPU system typically includes an operating system, a local cache operatively formed from processor internal memory, DASD-oriented external store, and storage protection (lock) and cache resource managers. Processes executing on a CPU generate read and write operations by way of the operating system. In turn, the read and write operations utilise the cache and lock resource managers to establish directory protectable access paths to pages currently resident in cache or as refreshed into cache from the external store.

"Virtual storage" involves the addressing of a storage space much larger than that available in the internal storage of a CPU. CPU processes tend to reference storage in nonuniform, highly localised patterns, making it possible for a small amount of real storage, properly managed to provide effective access to a much larger amount of virtual storage. If the referenced data is not available in internal storage, then new pages are swapped in from external storage, a process referred to as 'paging'.

The capacity of the system to manage pages is determined largely by the number of slots or "page frames" set aside in internal store for paging. If the sum of the subsets of pages referenced by processes exceeds the number of page frames in internal storage then at some stage it will be necessary to access external storage, a requirement known as faulting.

In a large data processing system a number of processes are typically running at any given time. Each process references its own subsets of pages, conventionally under the control of a part of the operating system referred to as a virtual demand paging system. In such a system the pages are usually stored in an associative store enabling a process to identify and access a desired subset by an associative tag or filename.

Ambiguity arises in such a system where two different filenames are used to access the same physical page. These "synonyms" are wasteful of cache space and create a cache invalidate problem since the cache manager usually has no way of associating the many possible names for the same data.

This difficulty is addressed in U.S. Patent 4,612,612 entitled "Virtually Addressed Cache", issued September 16, 1986, by treating the cach as

a virtual addressable entity. However, since in the system described in thus patent pages are cached by their virtual addresses, each page is treated independently, even though two virtual page addresses may reference the same ultimate address in real storage. Also, there is a finite possibility in such a system that virtual addresses in different address spaces are mappable to the same real address.

It is accordingly an object of this invention to provide an impaired arrangement for managing the access to pages mapped into a very large virtual external address space through a cache which effectively reduces the problem presented by synonyms.

According to the invention we provide a method for accessing data in a data processing system having a processor, internal storage organised as a data cache formed from addressable pages, and external storage addressable to access multiple pages associated in files, the method comprising the steps of referencing pages in a given file according to their addresses in a linear space as mapped into a virtual external storage address (VESA) and then as mapped into a physical address in external storage, and writing referenced pages into the cache using their VESA addresses as indexing arguments if not otherwise located in said cache, and, in response to a write request from said processor, updating in place those cached pages common to two files, otherwise shadow copying updated pages into another cache location using another VESA address.

We further provide a data processing system comprising a processor having internal storage formed from RAM-addressable pages and external storage formed from DASD-addressable pages, characterised by a cache adapted to assign device independent locations in a logical external storage space (VF0), (VF1) to pages (AV1P0, AV1P1) to be accessed, to assign, in response to an update (AV2P0, AV2P1') of a page not common to two files of associated pages, a further logical external storage space (VF2 for AV2P1') and, in response to an update of a page common to two files, to update such page in place in the cache, or otherwise to write a shadow copy thereof in cache assigning yet another logical external storage space (VF2') thereto.

In contrast to the aforementioned US patent No. 4,612,612, the invention uses an additional layer of indirection (VESA), i.e., pages are indexed in cache by their VESA arguments, avoiding synonym conflict. This requir s mapping to external storag via logical to VESA and VESA to r al.

Advantageously, the method of this inv ntion (a) generates a unique name for caching and avoids synonymy; (b) uses a unique name for locking; (c) stores data in cache and writes it out only upon change; and (d) if location of a page in real storage changes, then the cache is not invalidated because the logical address remains the same (invariant), and (e) physical backing for the virtual file is not required.

Brief Description of the Drawing

Fig. 1 sets out the organisation of storage in relation to a large main frame CPU.

Fig. 2 conceptually depicts virtual-to-real address translation, associative memory assist, and cache placement according to the prior art.

Fig. 3 shows a concept to virtual caching according to the prior art.

Fig. 4 depicts software caching and its placement according to the invention.

Fig. 5 sets forth the manner by which the synonym problem is avoided using the VESA-ordered pages in a cache according to the invention.

Fig. 6 illustrates updates in place and shadow copying.

Fig. 7 is another mapping example involving several different views according to the invention.

The invention can be conveniently practised in a general purpose computer such as an IBM/360 or 370 architected CPU having an IBM MVS operating system. An IBM/360 architected CPU is fully described in Amdahl et al., U.S. Patent 3,400,371, "Data Processing System", issued September 3, 1968.

A typical MVS operating system is described in IBM publication GC28-1150, "MVS/Extended Architecture System Programming Library: System Macros and Facilities", Vol. 1.

In this description the term 'page' will be used to designate a block of bytes. The number of bytes in a page is typically 4096.

Fig. 1 shows the relationship of organised storage to the central processing unit 1 (CPU) in a computer such as that referenced above. CPU 1 can access both internal storage 3 and external storage 5 over paths 11 and 13. Internal storage 3 includes processor storage 2, whose contents are byte addressable and randomly accessible, and expanded storage 4, whose contents are page addressable and randomly accessible. External storage 5 comprises one or more DASDs and stores multiple pages of information referenced by applications executing on CPU 1.

Typically, an application invoking the CPU processor references a page by either its virtual/linear or real space address to a cache 9. Cache 9 may be hardware or software implemented. If software implem nted, the cache could be located anywhere in internal storage 3. If the page is not available in cache 9, then either expanded storage 4 or xter-

nal storage 5 needs to be accessed.

Where multiple pages are accessed across the I/O boundary 7 in external storage, they may be processed according to methods as set forth, in Luiz et al., U.S. Patent 4,207,609, "Path Independent Device Reservation and Reconnection in a Multi-CPU and Shared Device Access System", issued June 10, 1980. When an access is made to internal storage, the processor waits until the access is completed. When access is made across the I/O boundary, the processor invokes another task or process while awaiting fetch (access) completion.

Address Translation and Cache Placement

Referring now to Fig. 2, there is conceptually depicted virtual-to-real address translation, associative memory assist, and cache placement according to the prior art.

As shown in Fig. 2, row (1), the conversion of a virtual address to a real address is usually implemented in hardware or fast microcode and involves an address translation or mapping. In a typical IBM System/370 machine, the address translation mechanism will decompose a virtual address into a page address and a relative page displacement.

As previously mentioned internal storage set aside in support of paging is organised into fixed locations called page frames. A page table may be used to correlate a virtual address reference in a program and the real address of a page frame in internal storage. The effective page address can be ascertained by adding the relative address to the page frame location. A further discussion may be found in Lorin and Deitel, "Operating Systems", The Systems Programming Series, copyright 1981 by Addison-Wesley Publishing Co., chapter 14 describing virtual storage, pp. 293-314.

Referring now to Fig. 2, row (2), there is shown one prior art technique for expediting the virtual-to-real address translation through the use of a "translation lookaside buffer" (TLB). The TLB 15 is formed from random access memory and is operative as an LRU associative memory in which the address of data being accessed is performed in parallel with the instruction being decoded by the CPU.

If a real cache 17 is placed ahead of real CPU main memory 19, as shown for instance in Fig. 2, row (3), then it has the advantage of storing pages with differ nt virtual addresses and pag s locat d in diff r nt virtual addr ss spaces. Howev r, it suffers the disadvantag that cache accessing occurs only aft r th virtual-to-r al translation has b n p rformed. In a real cache, address translation is first perform d followed by a tabl lookup.

Fig. 3 shows the plac ment of a virtual cach 21 prior to the address translation and real internal storage 19, an arrangement embodying principles found in US Patent No. 4,612,612, identified above.

As pointed out in that patent, at Col. 2, lines 43-49

"The buffer typically contains a small fraction of the main store data at any time. In the virtually addressed buffer, the location of the data is not a function of main store real addresses, but is a function of the virtual addresses. Therefore, main store addresses do not map to unique buffer addresses. More than one real address can be translated to the same virtual address location in the buffer."

The solution proposed in this patent is summarised at Col. 2, line 62, through Col. 3, line 2:

"Since different virtual addresses may specify the same data location that corresponds to a single real address location in main-store, it is possible that the virtual-address buffer will store more than one copy, called a synonym, of the same data at different locations. For this reason, a real-to-virtual translator translates main store real addresses to all buffer virtual addresses to locate buffer resident synonyms when modified data is stored into the buffer."

In the method of this invention, the cache is a software created and managed portion of internal storage. It serves as an external storage cache. In this regard, such a software cache is operatively different from the CPU cache hardware arrangement described in the Woffinden patent. In Woffinden's CPU cache, address resolution and access are in terms of microseconds, whereas resolution and access in the software external storage cache are in terms of milliseconds. This permits additional or refined processing.

Referring now to Fig. 4, there is depicted software caching and its placement according to the invention. Two address translations or levels of indirection are shown. The first is the virtual or logical address 23 mapped into a virtual external storage address space (VESA) 25, while the second is VESA mapped into the real external storage address space 5. Access to the cache 27 is only by way of a VESA argument. Cache 27 is positioned subsequent to the first mapping and prior to the second.

Avoidance of Synonymy Using VESA Ordered Pairs

Th use of two levels of indirection in th method of this invention takes advantage of the natur of base plus displacement addressing as described in conn ction with demand paging and virtual addressing. In this r gard, suppose an application executing on CPU 1 specifies the 100th

relative page. If there are multipl versions of that page, then each version has the same logical address. These are different versions of the sam file.

In this invention, the mapping from the name space to the intermediary space is many to one. Thus, two linear spaces sharing the same page would map to one single virtual external storage address (VESA) without synonym problems. The use of intermediate external storage avoids the synonym problem.

Referring now to Fig. 5, there are set out two versions of the same file and the virtual external cache 27. Illustratively, the first file 29 bears the logical name File A Version 1 (AV1). It comprises original pages 0 and 1. The second file 31 bears the logical name File A Version 2 (AV2). AV2 includes original page 0 and modified page 1 (page 1'). The pages 0, 1, and 1' are mapped into the VESA addresses (so-called virtual frames) VF0, VF1, and VF2, respectively. Only one copy of page 0 need be stored in cache 27.

Updating in Place and Shadow Copying

The method of the invention provides that responsive to a write in place of a page common to the original and updated files, the common pages are updated in place. This renders updated values available to both files (views). If the page to be written or updated is not common, then a shadow copy is written thereof in cache assigning yet another logical external storage space thereto.

Referring now to Fig. 6, there are shown changed files AV1 and AV2 and a different storage mix in cache 27. More particularly, assume that AV1 includes an original page 0 and an original page 1. Also, assume that AV2 consists of an original page 0 and a modified page 1'. The manager for cache 27 assigns VESA address VF0 to page 0, VF1 to page 1, and VF2' to page 1'. In the event an application updates page 0, then an update in place at VF0 will occur because page 0 is common to AV1 and AV2. However, an update to page 1' will be processed by way of writing the changed page 1" to a VESA address in the cache at VF2" and leaving the old page 1' as the shadow at cache VESA address VF2'.

Algorithmic Expression of the Method Using Another Example

Referring now to Fig. 7, th re is shown the double mapping of pag s from fil s 1 and 2 to VESA-ord red cache to real internal or external storage. Consider the following:

Suppose the initial state of the system consisted of file 1 formed from pages 1 and 2. Also, pages 1 and 2 are mapped into VESA addresses VF12 and VF40, and then mapped into real addresses R2 and R96. Next, assume that fil 2 was created initially as an image of file 1. The first concordance for pages 1 and 2 includes VF12 and VF40. The second concordance includes R2 and R96.

In order for page 2 of file 2 to become updated without sharing it with file 1, it is necessary to first allocate a new VESA, i.e., VF76 in cache 27. The concordance or page map for file 2 is then altered. After this, updated page 2' is written to cache location VF76. Real storage in the form of a DASD location, i.e., R102, is allocated and page 2' is copied therein. Parenthetically, VF40 remains the shadow location to VF76.

If page 1 is rewritten, then no new allocation in cache 27 is needed because the page is shared between the files. The existing mapping to VF12 remains the same and the updated page 1' is written therein. Likewise, the contents of VF12 are copied to DASD real location R2.

If the changes to cache 27 can be batched to the point where the cache is filled, then the transfer to DASD real storage can be at one time. This yields a transfer efficiency when compared to multiple discrete backing store updates.

If file 1 is deleted before it is written (e.g., if it is a temporary file), then none of its constituent virtual frames are ever allocated in real storage. Allocation to real storage is only required when frames are actually written: their existence in the cache does not require this.

Claims

1. A method for accessing data in a data processing system having a processor, internal storage organised as a data cache formed from addressable pages, and external storage addressable to access multiple pages associated in files, the method comprising the steps of

   referencing pages in a given file according to their addresses in a linear space as mapped into a virtual external storage address (VESA) and then as mapped into a physical address in external storage, and writing referenced pages into the cache using their VESA addresses as indexing arguments if not otherwise located in said cache, and, in response to a write request from said processing updating in plac those cached pages common to two fil s, oth rwise shadow copying updated pages into another cache location using another VESA address.

2. A method as claimed in claim 1, including the step of writing the pages out from the data

cache to physical addresses in external storage only upon change.

3. A method as claimed in claim 1 or claim 2 in which the validity of a page in the cache is maintained as long as the VESA address remains unchanged.

4. A data processing system comprising a processor (1) having internal storage (3) formed from RAM-addressable pages and external storage (5) formed from DASD-addressable pages, characterised by a cache (2) adapted to assign device independent location in a logical external storage space (VF0, VF1) to pages (AV1P0, AV1P1) to be accessed,

to assign, in response to an update (AV2P0, AV2P1') of a page not common to two files of associated pages, a further logical external storage space (VF2 for AV2P1') and, in

response to an update of a page common to two files, to update such page in place in the cache, or otherwise to write a shadow copy thereof assigning yet another logical external storage space (VF2') thereto.

5. A system as claimed in claim 4, wherein said cache is formed from non-volatile storage and is further adapted, in response to an indication that the cache is full, to allocate space in external storage to updated pages, copy updated pages to that space, and form a concordance between the logical external space location and the physical location in external storage.

6. A system as claimed in claim 5, in which the cache is formed from bistable remnant magnetic material or bistable battery backed electrostatic material.

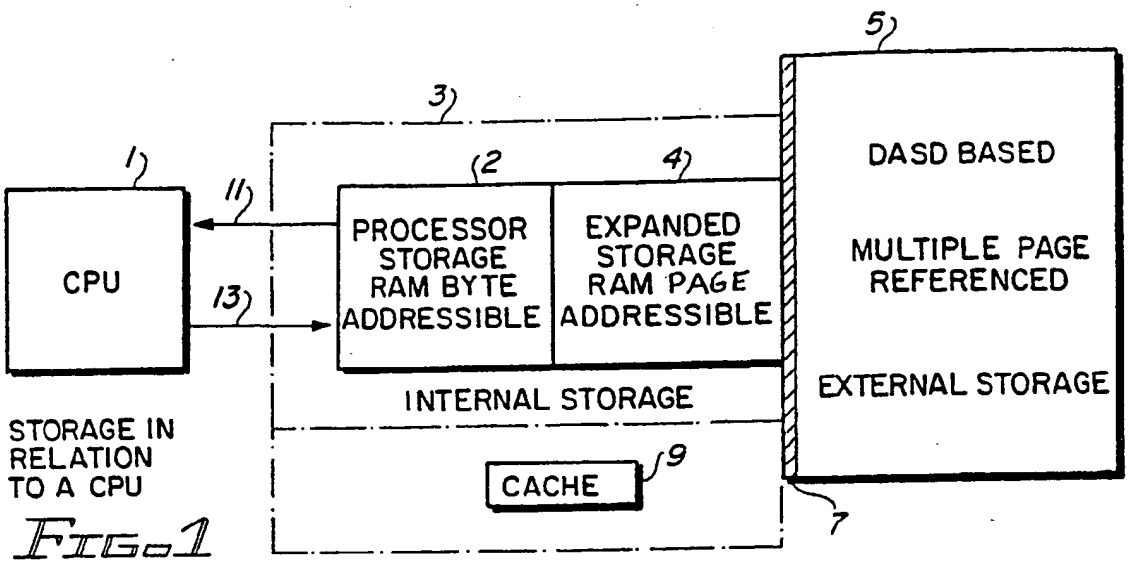
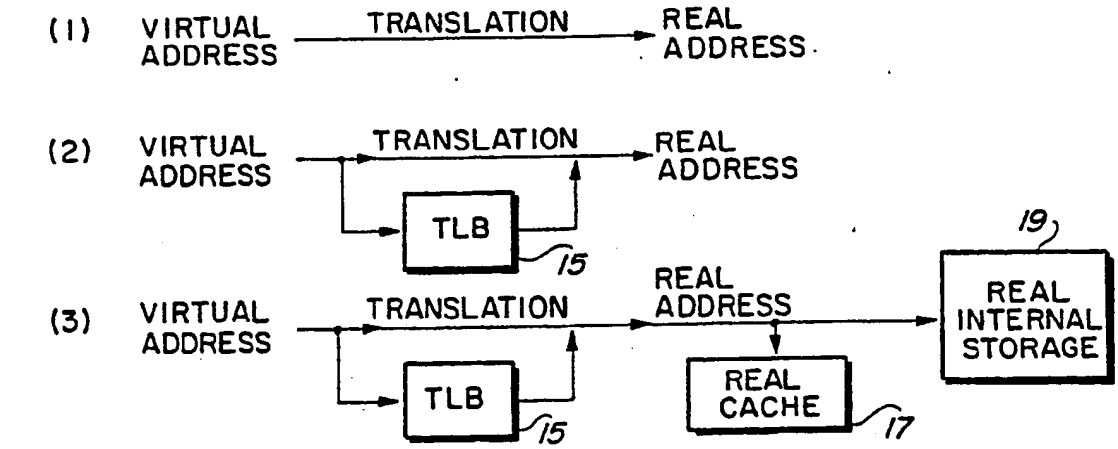
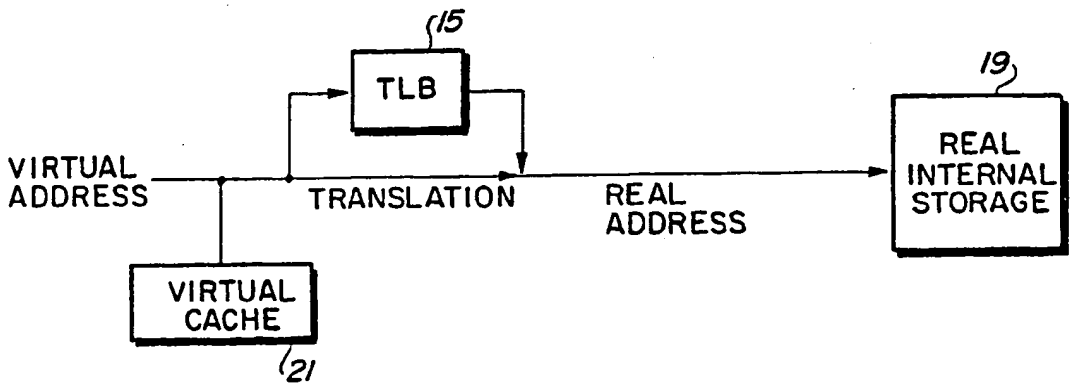


FIG-1



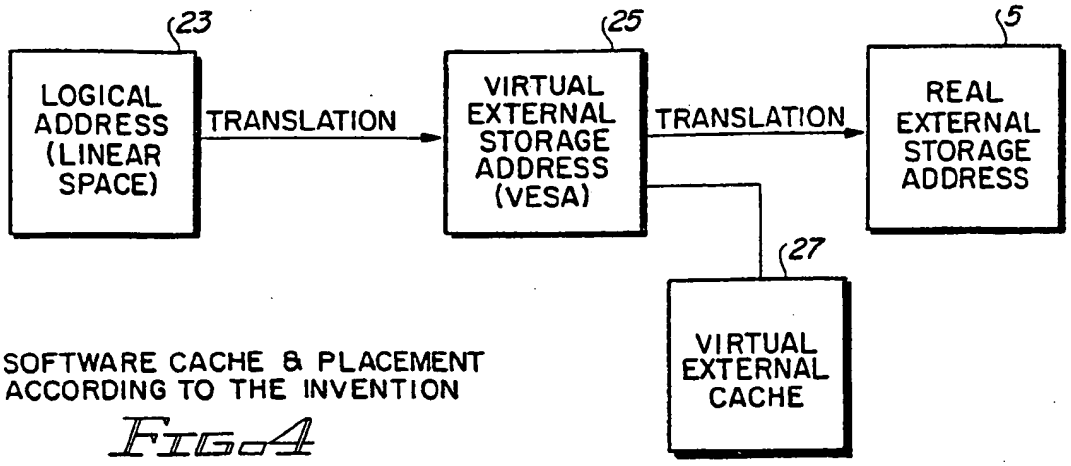
VIRTUAL TO REAL TRANSLATION & CACHE PLACEMENT

FIG-2



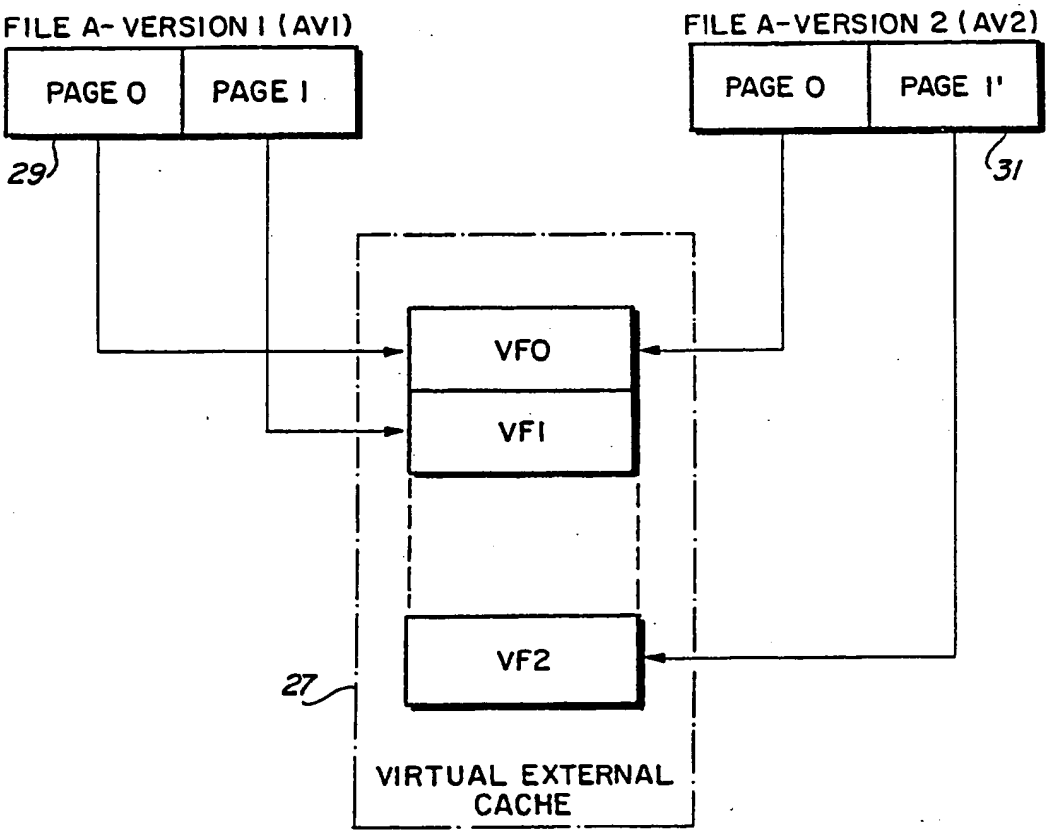
VIRTUAL CACHING (PRIOR ART)

FIG-3



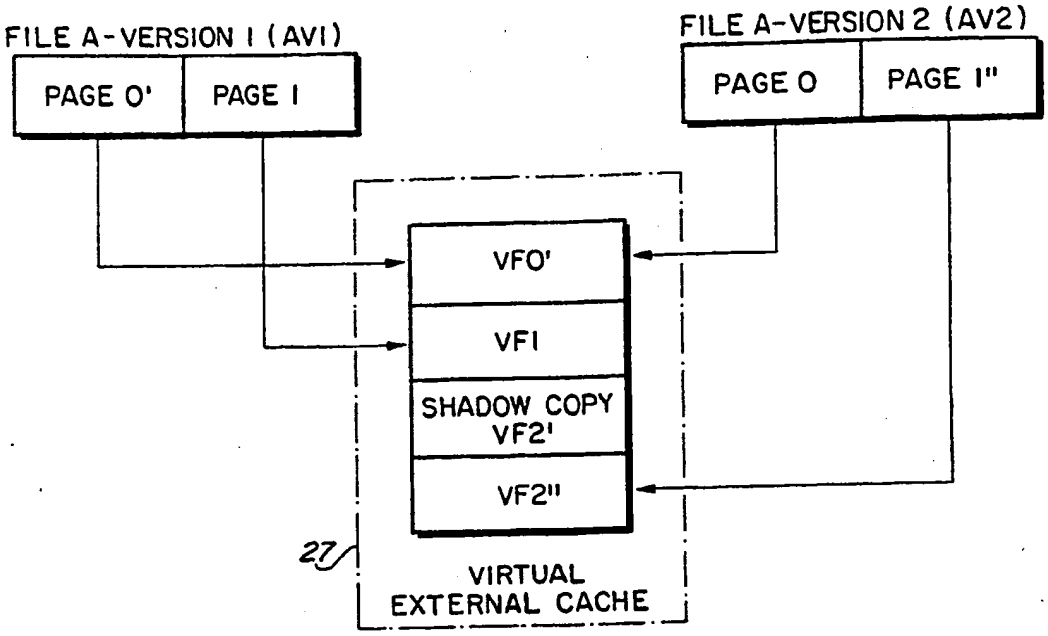
SOFTWARE CACHE & PLACEMENT ACCORDING TO THE INVENTION

FIG. 4



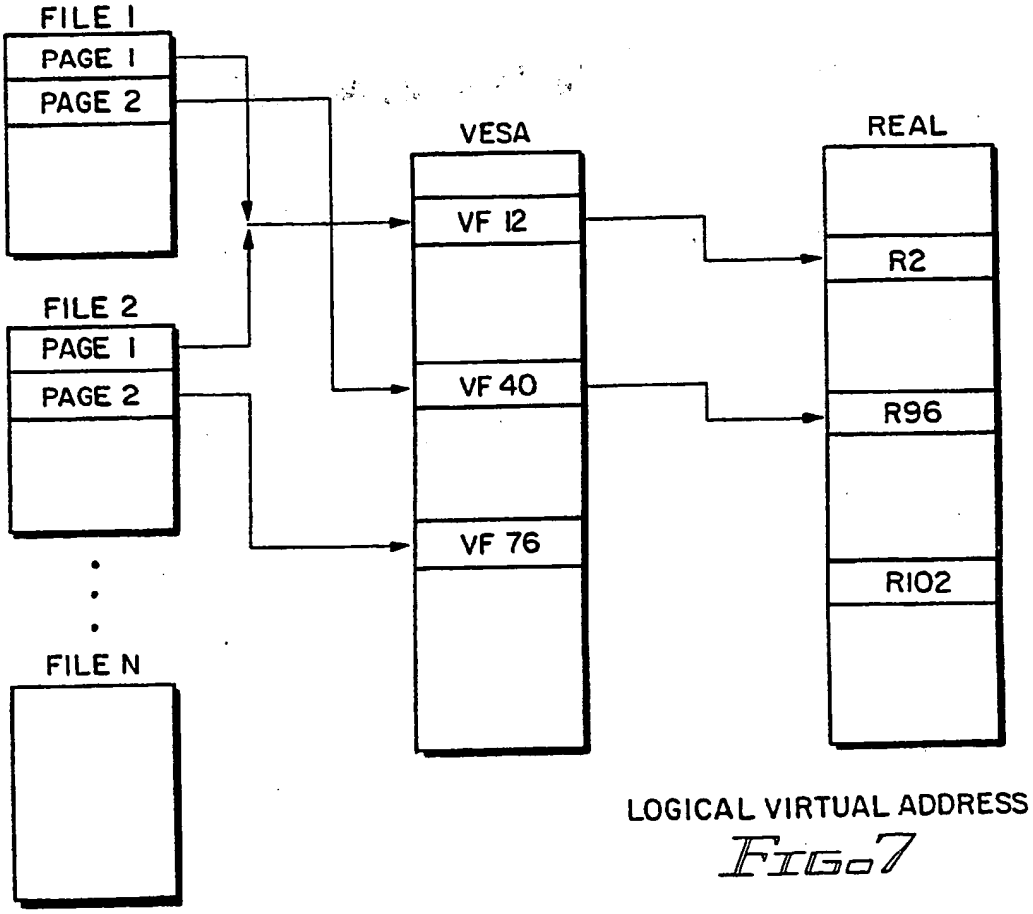
AVOIDANCE OF SYNONYM USING VESA ORDERED PAGES

FIG. 5



UPDATE IN PLACE & SHADOW COPYING PER INVENTION

FIG. 6



LOGICAL VIRTUAL ADDRESS

FIG. 7

(19) Europäisches Patentamt
European Patent Offic
Office uropéen des brevets

(11) Publication number: **0 441 508 A3**

(12) **EUROPEAN PATENT APPLICATION**

(21) Application number: **91300595.5**

(51) Int. Cl.5: **G06F 12/08**

(22) Date of filing: **25.01.91**

(30) Priority: **09.02.90 US 477704**

(43) Date of publication of application:
14.08.91 Bulletin 91/33

(84) Designated Contracting States:
DE FR GB

(88) Date of deferred publication of the search report:
21.04.93 Bulletin 93/16

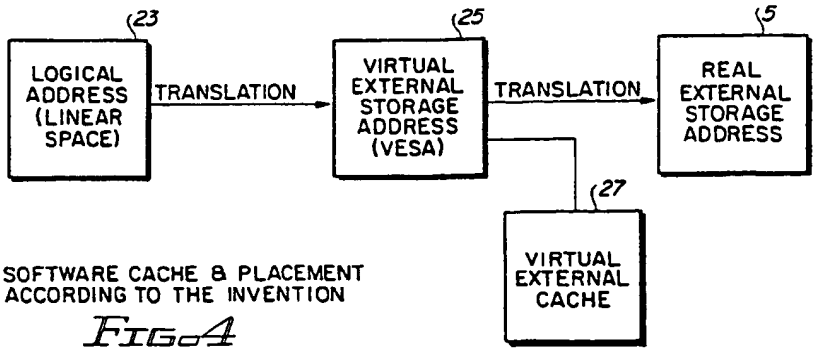
(71) Applicant: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: **Bozman, Gerald Parks**
609 Ramapo Valley Road
Oakland, New Jersey 07436(US)
Inventor: **Eisenberger, George**
54 Albemarle Road
White Plains, New York 10605(US)
Inventor: **Lett, Alexander Stafford**
402 MacGregor Drive
Mahopac, New York 10541(US)
Inventor: **Myers, James Joseph**
950 Columbus Avenue No.6
San Francisco, CA 94133(US)
Inventor: **Tetzlaff, William Harold**
37 Fox Den Road
Mount Kisco, New York 10549(US)
Inventor: **Unger, Jay Harold**
Knollwood Court
Mohegan Lake, New York 10547(US)

(74) Representative: **Killgren, Neil Arthur**
IBM United Kingdom Limited Intellectual Property Department Hursley Park
Winchester Hampshire SO21 2JN (GB)

(54) **Data storage using a cache.**

(57) Data in pages is mapped into a very large virtual external address space (25) through a cache without disturbing the logical view of the data and without having to assign physical or real backing store to said logical view. A data cache (27) is used in which pages are indexed according to a logical address (23) intermediate to their virtual address and their physical location in external storage (5). Pages common to two or more files are updated in place in the cache, while pages bound to only one file are shadow copied.



SOFTWARE CACHE & PLACEMENT ACCORDING TO THE INVENTION

FIG.4

EP 0 441 508 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 91 30 0595

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
P,A	PATENT ABSTRACTS OF JAPAN vol. 14, no. 340 (P-1080)23 July 1990 & JP-A-21 16 940 (FUJITSU) 1 May 1990 * abstract *	1,4	G06F12/08
A	PATENT ABSTRACTS OF JAPAN vol. 12, no. 42 (P-663)6 February 1988 & JP-A-62 189 543 (HITACHI) 19 August 1987 * abstract *	1,4	
A	US-A-4 459 658 (GABBE ET AL.) * abstract; claims *	1,4	
D,A	US-A-4 612 612 (WOFFINDEN ET AL.) * the whole document *	1,4	
A	US-A-4 757 447 (WOFFINDEN) 12 July 1988 * the whole document *	1,4	
A	EP-A-0 052 370 (HITACHI) * abstract; claims *	1,4	TECHNICAL FIELDS SEARCHED (Int. Cl.5) G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	23 FEBRUARY 1993	PFITZINGER E.E.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P0401)